Modification of the Algorithm for Beat Tracking of a Musical Melody

M. Yu. Khachai[a,b,c], K. S. Kobylkin[a,b], and D. M. Khachai[b]

[a] *Krasovsky Institute of Mathematics and Mechanics, Ural Branch, Russian Academy of Sciences,*
[b] *Ural Federal University Yekaterinburg, Russia*
[c] *Omsk State Technical University, Omsk, Russia*
e-mail: mkhachay@imm.uran.ru, kobylkin@imm.uran.ru

Abstract—A new efficient modification of the known heuristic algorithm for real-time beat tracking is proposed. An improved formula for updating the relative frequencies of time intervals between adjacent onsets is used in the modification. The algorithm has shown good performance on the MIREX Beat Tracking test base.

Keywords: beat tracking, onset detection, note attack.

DOI: 10.1134/S1054661813010069

INTRODUCTION

Design of computer algorithms for beat tracking of melodies is one of well-known problems of analysis and processing of digital audio. Its application scope includes synchronization of computer animation, sound-to-light electronic devices with played track, and analysis of musical compositions for automatic indexing to facilitate search in large databases.

The notion of the *beat* cannot be rigorously defined mathematically, since it is based on the intuitive perception of music by human ear. We can assume that beats are a sequence of strokes of an imaginary metronome, coordinated at every instant with a musical pattern of played track. One of the main requirements for beat tracking algorithm is resistance to noise and variations in performance conditions of a melody (volume, tempo, etc.).

The considered problem of the real-time processing of melody, which involves detection of beats based on relatively short history, is much more complex task theoretically than offline processing, when the entire melody is available.

The most well-known beat tracking algorithms are presented in [1–4]. In this paper, we propose a simple heuristic algorithm for detection of beats in musical melody using the ideas from [5]. The algorithm is implemented in Java using the Java Media Framework 2.0 library. The developed application is used to synchronize computer animation with played track. The comparative testing of the algorithm based on data from the MIREX Beat Tracking Contest [10], which includes 20 tunes of different musical genres, gives an average cross-correlation of 0.34, which is quite a good result.

DEFINITIONS

A piece of music is a structured set of musical sounds (notes) of a certain rhythmic value. Rhythmic pattern is one of its components. The rhythmic pattern of any composition is determined by the particular points in time, *beats*, that are accentuated by performer using volume and/or intonation. Generally they split the performance duration of a piece of music into equal intervals, called *parts of a measure*. The first part of the measure is called the *main* one. Beats are also characterized by a metrical level. For example, for the 4/4 measure, levels of quarter, half, or whole notes are possible. *Tempo* is the speed with which beats occur during performance. It is measured in beats per minute. In Western music, tempos of 40 to 240 beats per minute are permissible.

The following concepts are based on the representation of a note in the form of the amplitude of the sound signal, which is a function of time.

Definition 1. Note *attack* [6] is a short time period of a strong increase in the amplitude envelope. *Transient* is a short time interval including the attack during which the signal changes very quickly in some unpredictable way. *Onset* is the beginning of transient.

Onset detection is important task for transcription of a piece of music. It is also important in tracking of beats since they tend to coincide with the onsets of "bright" events of a piece of music [7].

DESCRIPTION OF THE ALGORITHM

The proposed modification of the algorithm operates according to the following general scheme [8]. At the first stage, onsets are calculated for the source signal being processed in real time (linear 16-bit stereo sound with the 44.1 kHz sampling frequency); some history of these times is kept.

At the second stage, the length of the time interval $\Delta_0 = t_n - t_{n-1}$ between the last detected onset t_n and the previous onset t_{n-1} is calculated. A certain range of rhythmic values (periods) corresponds to the range of possible musical tempos with rates from the fastest to slowest ones. Rhythmic values Δ_0, which do not belong

Received March 20, 2011

to this range, are discarded by the algorithm as not corresponding to any known musical tempos. The algorithm computes relative frequency for each rhythmic value of the range. Prior to start of melody playback, the frequencies of all rhythmic values are assumed equal. The relative frequency of the last observed rhythmic value Δ_0, and all those rhythmic values close to it, increases, and the frequencies corresponding to the rest of the rhythmic values are reduced. At the time when the rhythmic value Δ_0 is close to the rhythmic value with the highest frequency, the value of $1/\Delta_0$ is treated as a new estimate of the tempo.

The software algorithm is implemented in a separate computational thread. In the parallel thread, the so-called metronome algorithm is running. It gives estimates of beats, separated from each other at regular intervals in accordance with the current estimate of the tempo. These two algorithms interact with each other in the following way: if the last observed rhythmic value Δ_0 is close to the rhythmic value with the highest frequency and the current tempo is different from $1/\Delta_0$, the metronome will start to beat at regular time intervals Δ_0 starting from the t_n moment as a beat.

Onset detection. Onsets corresponding to "bright" events of musical melody can be found by marking (see, e.g., [6]) jumps of audio signal energy (as a function of time) in multiple frequency bands. The energy function is the sum (in a range of frequencies) of squares of modules of samples of the fast Fourier transform of the original audio signal applied with a sliding window. With a larger number of frequency bands, a better onset segregation is achieved, but the amount of memory and processing time used increase. The window length, being too large, ensures a smooth change of the energy function, but will make it impossible to find some onsets, and, conversely, in the case of a small window, the search for onsets with help of energy functions will be complicated by presence of a large number of jumps not corresponding to rhythmic events. The authors chose the logarithmic scale of 64 frequency bands and the window length of 1024, i.e., about 25 ms.

Suppose that during the processing a block of 1024 samples of the original signal was obtained, and for each frequency band energy function $\{E_i\}$ was calculated. Some history of the ith range energy values is kept, and the average energy M_i, $i = 1, \ldots, 64$ can be calculated. The algorithm detects the onset in the block if the inequality $E_i > CM_i$ holds true for some $i = i_0$, where C is the threshold value, which is characteristic of the algorithm for determining onsets, and M_i is the average over 43 consecutive blocks.

Update of frequencies. A listener is inclined to consider similar those tempos whose ratio is some power of two. Therefore, in order to improve the stability of recognition, it is convenient to identify rhythmic values whose ratio is 2^k, where k is an integer, by adding up their frequencies, and to calculate the relative frequencies for rhythmic values in a narrower range. In the range of rhythmic values that lie between 0.28 and 2.21 s, we selected the subrange **I** of rhythmic values from 0.55 to 1.09 s.

Let us introduce a distance function g between two rhythmic values Δ_1 and Δ_2 of this subrange

$$g(\Delta_1, \Delta_2) = \min\{M - m, 2m - M\}, \quad (1)$$

where $m = \min\{\Delta_1, \Delta_2\}$ and $M = \max\{\Delta_1, \Delta_2\}$.

Definition 2. We consider rhythmic values Δ_1 and Δ_2 *close*, if

$$g(\Delta_1, \Delta_2) < \gamma \quad (2)$$

for some small γ.

The formula for updating frequencies of rhythmic values of the range in the determination of the next period Δ_0 is similar to that in [5]

$$p(\Delta) = \alpha p(\Delta) + \beta e^{-g^2(\Delta, \bar{\Delta}_0)}, \quad (3)$$

where $\Delta \in \mathbf{I}$, $\Delta_0 = t_n - t_{n-1}$, $\alpha > 0$, $\beta > 0$ and $\bar{\Delta}_0 = x\Delta_0 \in \mathbf{I}$, $x = 2^t$ for some integer t. After recalculation of frequencies of rhythmic values by (3), the values $p(\Delta)$ are normalized so that their sum is 1.

RESULTS AND DISCUSSION

We carried out an experiment using the MIREX Audio Beat Tracking test collection. The collection consists of twenty melodies lasting 30 s and belonging to different musical styles. Beats for each melody are marked by 40 music experts. In order to evaluate the performance of the algorithm, the McKinney method was used [9]. For beat times (in seconds), generated by the algorithm, as well as for those labeled by experts, a binary sequence is formed. These sequences with the same duration of 30 s and the frequency of 100 Hz contain unit pulses in beat times. Let us denote the binary sequence for a melody by $y(n)$ (respectively, by $a_s(n)$) formed on the basis of the output sequence generated by the algorithm (accordingly given by the sth expert). The average cross-correlation between y and a_s with the center at 0 (and some window) over all the experts gives the performance of the algorithm P for a particular melody

$$P = \frac{1}{S}\sum_{s=1}^{S}\frac{1}{K}\sum_{m=-W}^{W}\sum_{\substack{n \\ n-m\geq 1, \\ n-m\leq N}} y(n)a_s(n-m), \quad (4)$$

where N is the length of the sequence a_s, $W - 1/5$ of the median distance between neighboring units in a_s, S is the number of experts, and

$$K = \max\left\{\sum_{n=1}^{N} y(n), \sum_{n=1}^{N} a_s(n)\right\}. \quad (5)$$

The mean value of P over all melodies shows the overall performance of the algorithm. The experimental results are given below.

Test results

Number	Musical style	Result
1	popular	0.26
2	rock	0.36
3	folk	0.29
4	rock	0.38
5	folk	0.34
6	popular	0.35
7	classic	0.39
8	popular	0.35
9	rock	0.37
10	choir	0.36
11	popular	0.35
12	classic	0.39
13	folk	0.27
14	popular	0.33
15	popular	0.36
16	classic	0.36
17	popular	0.37
18	rock	0.37
19	hard rock	0.28
20	folk	0.18

SIMPLEST MATHEMATICAL FORMULATION OF TEMPO DETECTION BY THE SEQUENCE OF ONSETS

The rhythmic pattern of a simple melody can be specified as a sequence $a(\cdot)$ of beats (in seconds) of length N such that its adjacent terms differ from each other by a certain number, which is approximately equal to T, corresponding to the melody tempo.

Let us consider Bernoulli scheme of N independent trials with probability $0 < p < 1$ of success in a separate trial. The registration of the "bright" event at the time $a(i)$ by the algorithm is regarded as a success. Let us denote by $z(\cdot)$ a random binary sequence of length N such that $z(i) = 1$ (respectively, $z(i) = 0$) in the case of success (respectively, failure) in the ith trial. We also consider the subsequence $b(\cdot)$ of the sequence $a(\cdot)$ including only those of its members whose numbers are unit positions in $z(\cdot)$.

Task. Find T using realization of the sequence $b(\cdot)$.

SOFTWARE IMPLEMENTATION OF THE ALGORITHM

The algorithm is implemented in Java using the Java Media Framework 2.0 (JMF) library, providing the developer with the unified architecture and messaging protocol for reading, processing, and display of media data. The library supports most of standard formats, i.e., AVI, MIDI, MPEG (MP3 and MP4), etc. It uses an abstract model based on the following key concepts.

Definition 3. The *data source* is an object-oriented wrapper over the data flow (video or audio). A *player* implements the algorithms of data representation (similar to video monitor or audio system). The *processor* is the component that implements more general algorithms of data transformation.

The most important advantage of the library is its extensibility, which is based on a system of plug-ins, and makes it possible not only to organize the interaction with new media devices, but also to incorporate easily new data transformation algorithms. This architecture is based on the Processor class (Fig. 1). It derives from the Player class, receives its input data in the form of the instance object of the DataSource class, and carries out user-defined transformations of these data. Data obtained as a result of the transformation can be transferred to the output of the device (such as a monitor) or sent for further processing.

Consider the following data transformations supported by the JMF library.

Definition 4. *Demultiplexing* is the process of initial analysis of input signal, in which it is divided into separate "tracks." For example, the traditional AVI format contains one video track and one or more audio tracks. *Encoding/decoding* is the format conversion of each of the selected tracks of the input signal. For example, the audio data stored in the mp3-file should be decoded before playback. The process of combining multiple tracks into a single output stream is called *mul-*

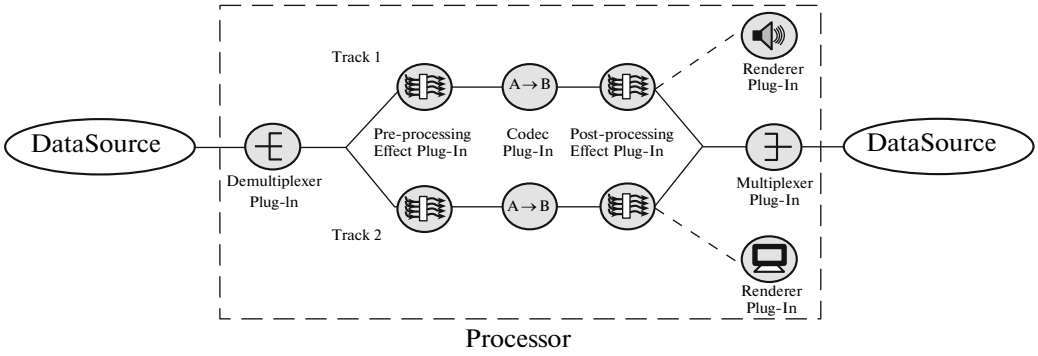

Fig. 1. Using the processor class.

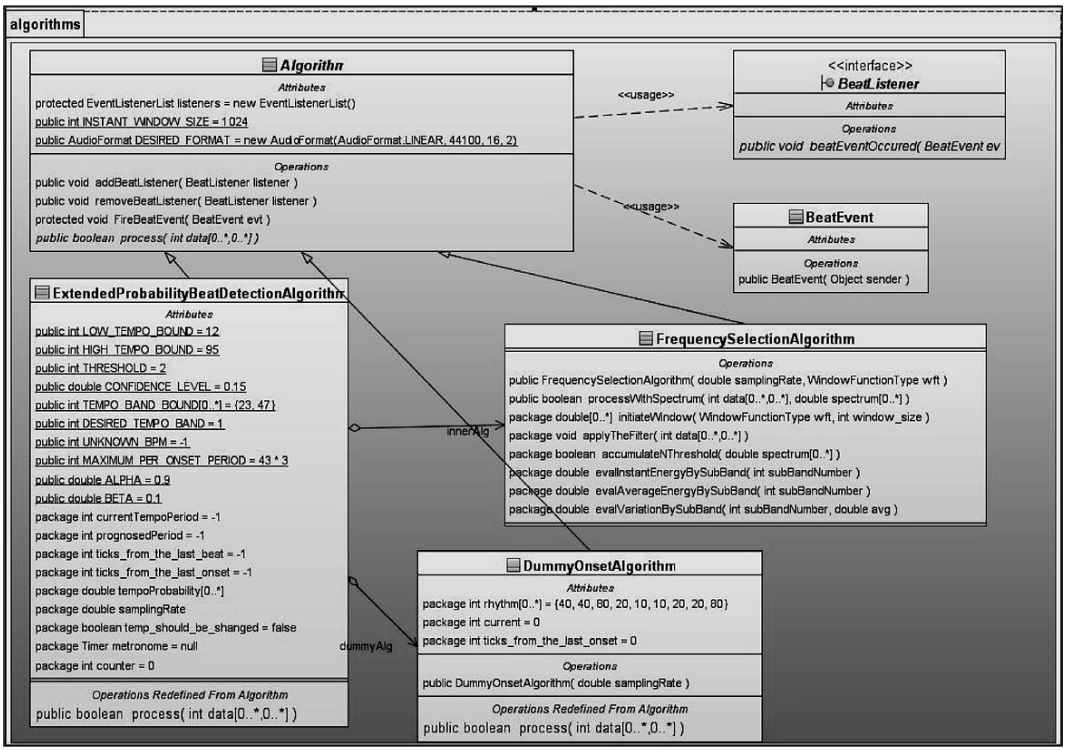

Fig. 2. Algorithms Package: class diagram.

tiplexing. Rendering is the process of submission of video and audio data to the user. Rendering of audio data often consists in their playback by the audio system.

Application description. The logically designed application consists of two subsystems, executed in parallel threads and interacting with each other asynchronously by message transfer. The first subsystem is responsible for interaction with the input device using JMF interfaces and processing of the digital audio input. An instance of the Processor class sequentially transfers the original data for processing to the instance of the developed BeatRenderer class, which implements the Renderer library interface, which, in turn, sends the buffered data to the instance of a class that inherits the Algorithm class and is hosted in the developed *rhythm.detection* library of the digital audio signal processing and beat tracking. Thus, the ExtendedProbabilityBeatDetectionAlgorithm class implements the above main algorithm that predicts the most likely tempo of performance and transfers the corresponding event to the caller and to the second subsystem. The second subsystem actually simulates a metronome, generating periodic signals corresponding to the estimated tempo.

For the sake of convenience of description of the structure of the application, we use class diagrams constructed in UML [11]. Consider the structure of each of the packages. The basis of the *rhythm.detection.algorithms* package is the interface BeatListener and base classes Algorithm and BeatEvent. The Algorithm class is the base class for each of the analysis algorithms implemented in the software, whether it is a beat tracking algorithm or algorithm for onset detection. Its main operation is the abstract method *process*, which is redefined in each of the child classes. It is used to implement specific algorithms and is called by an external program (Fig. 2).

The package contains implementations of modifications of the algorithms mentioned above. Thus, the classes ProbabilityBeatDetectionAlgorithm and ExtendedProbabilityBeatDetectionAlgorithm implement the basic beat tracking algorithm for music performed online on the basis of sequences of onsets detected by instances of the classes SimplestEnergyAlgorithm or FrequencySelectionAlgorithm.

Since the processing of the input signal is conducted asynchronously (in a parallel thread), each of the classes described above notifies the main program about the identification of an onset or beat by generating the corresponding event instance of an appropriate subclass of the class BeatEvent. For the external program to process these events, the interface BeatListener should be implemented in it. There should also be a subscription for the delivery of events using the Algorithm.addBeatListener method (Fig. 3). As mentioned above, the JMFHelpers package is responsible for the interaction with the JMF library. This package contains two classes CaptureNBeatController and BeatRenderer. The first class is used by the caller to initialize an instance of the Processor class and for association of an instance of the second class with it. The BeatRenderer class imple-

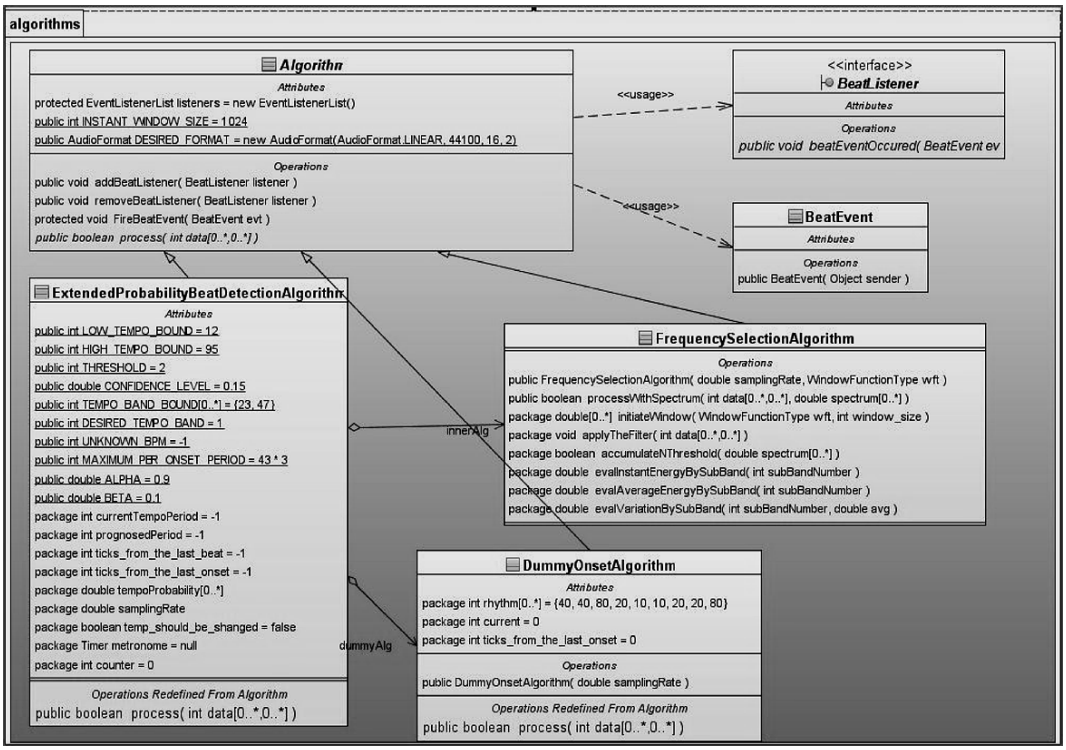

Fig. 3. JMF Helpers Package: class diagram.

ments the system interface Renderer, described in the JMF library. An instance of this class is responsible for buffering of the audio input signal and its transfer for processing by the instance of the Algorithm class, implementing one of the beat tracking algorithms.

CONCLUSIONS

A new modification of the well-known simple heuristic algorithm for real-time beat-tracking is proposed, which uses an improved formula for updating the relative frequencies of time intervals between adjacent onsets. The algorithm shows good performance on the MIREX Beat Tracking test base. Its Java-implementation is used to synchronize the music with computer animation.

ACKNOWLEDGMENTS

This work was supported in part by the Ural Branch, Russian Academy of Sciences (grant nos. 12-P-1-1016 and 12-S-1-1017/1) and the Russian Foundation for Basic Research (project nos. 10-01-00273 and 10-07-00134).

REFERENCES

1. S. Dixon, "Automatic Extraction of Tempo and Beat from Expressive Performances," J. New Music Res. **30**, 39–58 (2001).
2. E. Large and J. F. Kolen, "Resonance and the Perception of Musical Meter," Connection Sci. **6**, 177–208 (1994).
3. E. D. Scheirer, "Tempo and Beat Analysis of Acoustic Musical Signals," J. Acoust. Soc. Am. **103**, 588–601 (1998).
4. M. Goto and Y. Muraoka, "Real-time Rhythm Tracking for Drumless Audio Signals Chord Change Detection for Musical Decisions," in *Proc. Int. Joint Conf. on Artificial Intelligence. IJCAI'97 Workshop on Computational Auditory Scene Analysis* (Nagoya, 1997), pp. 135–144.
5. K. Jensen and T. H. Andersen, "Real Time Beat Estimation using Feature Extraction," in *Proc. Computer Music Modelling and Retrieval Symposium* (Springer Verlag, 2003), Vol. 2771, pp. 13–22.
6. J. P. Bello, L. Daudet, S. Abdallah, Ch. Duxbury, M. Davies, and M. Sandler, "A Tutorial on Onset Detection in Music Signals," IEEE Trans. Speech Audio Processing **13** (5), Part 2, 1035–1047 (2005).
7. H. Longuet-Higgins and C. Lee, "The Perception of Musical Rhythms," Perception **11**, 115–128 (1982).
8. P. Desain, "A (de)Composable Theory of Rhythm," Music Perception **9**, 439–454 (1992).
9. M. F. McKinney, D. Moelants, M. E. P. Davies, and A. Klapuri, "Evaluation of Audio Beat Tracking and Music Tempo Extraction Algorithms," J. New Music Res. **36** (1), 1–16 (2007).
10. MIREX Contest. www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking_Results.
11. C. Larman, *Applying UML and Patterns* (Prentice Hall, New Jersey, 2000).

Milkhail Yur'evich Khachai. Born in 1970 in Krasnoturyinsk, Sverdlovsk region. In 1993, graduated from the Faculty of Mathematics and Mechanics of Ural State University. In 1996 have got his PhD (candidate of sciences degree), and in 2005, his doctoral degree on "Discrete Mathematics and Mathematical Cybernetics." Since 1994 has worked at the Institute of Mathematics and Mechanics, Ural Branch, Russian Academy of Sciences. Since 2007, Head of the Department of Mathematical Programming. He is a specialist in mathematical learning theory and combinatorial optimization. He is the author of 83 scientific publications, of which 12 are in PRIA.

Konstantin Sergeevich Kobylkin. Born in 1977. In 2000 graduated with Master's degree from the Department of Mathematics and Mechanics, Ural State University. In 2005 got his PhD. degree on "Discrete Mathematics and Mathematical Cybernetics." Since 2001 has been working at the Krasovsky Institute of Mathematics and Mechanics, Ural Branch, Russian Academy of Sciences. Position: Senior researcher. Field of interest: data mining, pattern recognition theory, processing of audio signals. Author of 8 publications. Member of the RAROAI Association.

Daniil Mikhailovich Khachai. Born in 1995 in Yekaterinburg. Currently a student of physics and mathematics school no. 5. Research interests are related to the study and implementation of software algorithms to analyze audio signals. This article is his first scientific publication.